SecureZone: A Framework for WS – Secure Conversation and WS – Trust

Ruchith Fernando, Dimuthu Leelarathne, Malinda Kaushalye
Computer Science and Engineering Department,
University of Moratuwa, Sri Lanka
Tel: 94-11-2650301 Ext. 3100
Fax: +94-11-2650912

Abstract

"SecureZone" stands one step ahead of the existing technology available for web services security. It implements an area in "web services security" that is currently undergoing research. SecureZone has been accepted by the Apache Foundation, as a sub-project under the "wss4j"-Web Service Security for Java project. Therefore it has become the only widely known open-source implementation available that confirms to WS-Secure Conversation specification. It is expected that SecureZone would provide researchers an opportunity to evaluate and improve the WS-Secure Conversation and WS-Trust specifications. Since SecureZone demonstrates the role of WS-Trust in WS-Secure Conversation it will also be invaluable for studies carried out for resolving interoperability issues between the specifications.

There is no other platform independent implementation available for WS-Secure Conversation, and the web service community is expected benefit greatly from the attempt.

1. Introduction

Web services have emerged as the next generation of Web-based technology for exchanging information. Today there is a considerable amount of web service security-related research and activities occurring at the various standards organizations. SecureZone implements an area in web service security that is being currently researched.

The framework-SecureZone provides facilities for developers to enable secure conversation between two web service communication parties. The secure conversation implemented by the SecureZone conforms to WS-Secure Conversation and WS-Trust specifications.

WSE2.0 for Microsoft .NET is a supported add-on to Microsoft Visual Studio.NET that enables developers to build security-enhanced web services. It contains implementations for all published security specifications for web services. There was no other known implementation of WS – Secure Conversation, until SecureZone stepped-up to address the need. SecureZone provides a Java based solution which will be platform independent and by making it open-source it is expected to provide Java based web-services developer community the benefits of WS – Secure Conversation and WS - Trust.

The next section would introduce the background technologies and it will be followed by the architecture of the framework. Then results of the framework will be evaluated, and it will be compared with the existing technologies.

2. Background

The base specification of web service security protocol stack, namely WS-Security Specification describes mechanisms for securing SOAP messages on per message basis. WS-Security specification describes how to provide integrity, confidentiality and authentication of SOAP messages by using cryptography. In other words it specifies means of encryption and signatures for SOAP messages.

WS – Secure Conversation introduces a security context (referred to as Security Context Token or SCT), which can be used to authenticates a series of messages, in contrast to the per message security provided in WS-Security. SCT contains a session key and a secret. A primary goal of the WS – Secure Conversation specification is to define how SCTs are established.

After establishment the security context or the SCT will be shared among the communicating parties and will be used throughout the lifetime of the communication. The underlying concept of WS – Secure Conversation can be viewed as a secure "session" between two web service parties.

Based on the secret associated with the SCT, new keys for encryption and signature are derived, called "derived keys". A primary goal of the specification is to describe how derived keys are computed and key material is exchanged. Even though it requires additional communication and time for establishment of SCT, since derived keys are being used for providing security for subsequent messages, overall performance and security is expected to increase.

So why do we need WS-Secure Conversation? The WS-Security specification provides security on per message security and this is called message authentication model. The WS-Secure Conversation specification points out the weakness of this mechanism as; "The [WS-Security] specification focuses on the message authentication model. This approach, while useful in many situations, is subject to several forms of attack (see Security Considerations section of [WS-Security] specification). Accordingly, this specification introduces a security context and its usage. The context authentication model authenticates a series of messages thereby addressing these shortcomings, but requires additional communications if authentication happens prior to normal application exchanges."

It should be noted that WS – Secure Conversation by itself does not provide a complete security solution for Web services. WS – Secure Conversation is a building block that is used in conjunction with other Web service and application-specific protocols to accommodate a wide variety of security models. [1]

Purpose of SecureZone framework is to provide WS secure conversation but it also contains components of WS-Trust as well. Why? WS-Secure conversation requires WS-Trust to establish secure conversation in several situations.

If two communicating parties do not trust each other, they cannot establish a security context straight away. They need to determine if they can "trust" the asserted credentials of the other party. This is where WS-Trust comes to play. WS-Trust enables applications to have trusted SOAP message exchanges. Therefore by implementing STS (Security Token Service) as specified in the WS-Trust specification, it is possible to carry out Secure Conversation between parties in different trust domains. The secure conversation framework will be more comprehensive with components from WS-Trust.

3. Technologies Used

The SOAP engine is a framework for constructing and processing SOAP messages. Axis is a one of the most widely used SOAP engines by the Web Services community. AXIS (stands for Apache eXtensible Interaction System) is the most popular open source implementation of soap engine for java. Therefore it was decided to implement WS Secure conversation for Axis. Message processing in AXIS is built on a series of modules known as handlers. Hence adding custom functionality it's simply a matter of inserting a handler into the handler chain.

WS-Secure conversation uses the functionality provided by WS –Security, which is the base of the web services security specification stack as described above. Therefore an implementation of WS-Security was needed for the project. The WS-Security implementation available at Apache "wss4j" project was used for SecureZone.

4. Architecture of the framework

4.1 Architecture of WS-Secure Conversation

The major question was how to implement WS-Secure Conversation for Axis? It was decided that the best solution for this problem is to use Axis handlers. Axis handlers are modules used to process SOAP messages, as already explained above. Therefore the most natural way to implement WS-Secure conversation was using handlers. WS-Secure conversation is service specific functionality – hence "service handlers" had to be used for the purpose.

Furthermore a design decision was made to implement core components independently from Axis. There are 2 core components in the architecture – namely "WS- Secure Conversation element processing component" and "KeyDerivation component". These are designed to be independent from Axis.

Role of Axis Conversation handlers

Two Axis-Handlers are used to plug in the WS-Secure Conversation into Apache Axis – one at server side and the other at client side. In addition to interfacing to Apache Axis, Conversation Handlers play 4 major roles.

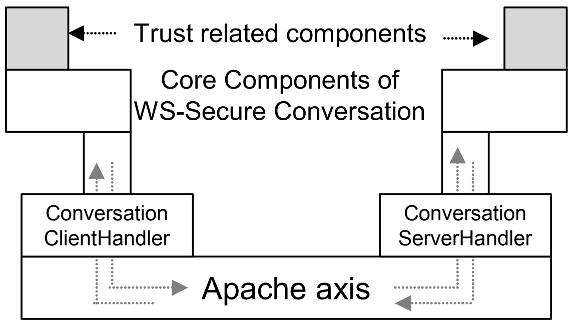
Trust related components
Core Components of WS-Secure Conversation
Conversation ClientHandler
Conversation ServerHandler
Apache axis

Figure 1: Interfacing to Apache Axis

- **Plug-in the WS-Secure Conversation functionality into Axis**

Handlers are the standard way provided for plugging new functionality into Axis. It doesn't require altering the Axis code – it is a simple matter of changing the configuration file.

If a developer wants WS-Secure Conversation to be used by the web service, it is a matter of changing the configuration file (*.wsdd).

- **Marshalling the core components to execute WS - Secure Conversation**

Handlers control the creation and processing of secure conversation SOAP requests. It employs the core components provided in our implementation (Eg: Key Derivation components) as required.

Therefore the developer is released from the responsibility of managing the processing of Secure Conversation SOAP headers.

- **Allow developers to configure WS – Secure Conversation parameters**

Secure conversation Handlers allow developers to control the way WS – Secure Conversation operate for their web service. When properly configured handlers control the way WS – Secure Conversation occurs.

The Axis deployment descriptor files (*.wsdd) can be used by the developer to specify all these parameters.

- **Manage and control WS-Trust requirements needed in secure conversation**

Secure Conversation between two parties can be initiated using a Security Context Token obtained by a Security Token Service (Figure 1).

If Secure Conversation requires such functionality, Conversation Handlers use the components provided by WS-Trust to communicate with a Security Token Service.

Roles of core components

Key Derivation component

Key derivation module will handle two main tasks

- Storing the information with respect to all the sessions
- Deriving keys as requested using the session information

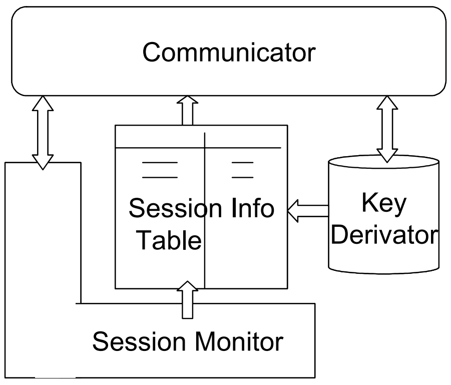

Figure 2: Key derivation module

The communicator is the interface to the key derivation module. It exposes necessary functionality for Conversation Engine and Conversation Manager to store session information and obtain derived keys. Session information table carries all the session information (i.e. security context token information and the derived key token information).

Key derivator is in charge of deriving key according to the given information using the session information table. Conversation session management is taken care of by the session monitor, i.e. security contexts are expired appropriately by session monitor. This component is independent of the SOAP Engine.

WS- Secure Conversation element processing component

The component consists of two classes – Conversation Manager and Conversation Engine. The purpose of the component is to create process and verify the validity of the elements used for secure conversation. Conversation Manager works at sender while Conversation Engine works at receiver.

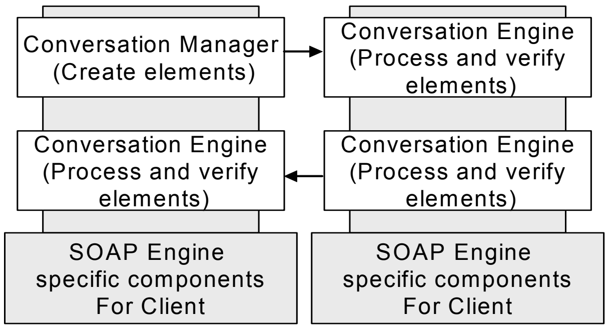
Conversation Manager (Create elements)
Conversation Engine (Process and verify elements)
Conversation Engine (Process and verify elements)
Conversation Engine (Process and verify elements)
SOAP Engine specific components For Client
SOAP Engine specific components For Client

Figure 3: Conversation Manager and Conversation Engine

Conversation Manager is in charge of creating secure conversation elements. It is capable of adding derived key tokens to the soap message, creating signatures using derived keys and encrypting using derived keys.

The Conversation Engine processes WS-Secure Conversation elements. It is capable of performing signature verification based on derived keys and performing encryption based on derived keys.

The Conversation Manager and Conversation Engine are designed and implemented to operate independent of Apache Axis. They are used appropriately by Handler classes.

4.2 Architecture of WS-Trust

WS-Trust implementation of SecureZone framework provides a mechanism to perform operations related to security tokens. The main idea of implementing WS-Trust in SecureZone is to assist the scenarios identified in WS – Secure Conversation. SecureZone framework uses a Security Token Service to achieve the above by providing following functionalities.

1. Issue security tokens.
2. Validate security tokens.
3. Renew security tokens.

The main objective of above functionalities is to establish a trust relationship between parties. To assist this, SecureZone implementation provides a Security Token Service (STS) as specified in WS-Trust specification [1] along with other supporting components.

STS manager and workers

While designing the Trust architecture a special effort was made to separate the tasks that are axis specific from that are not. Security Token Service Manager (STS Manager) is the decision making component of the Server Side of the security token service. STS Manager identifies the request type by inspecting the request and handover it to an appropriate worker. To assist this process, the Request Resolver is used.

A worker can be an issuer, renewer or a validator. The implantation provides interfaces for each of these and any application developer can use these interfaces to write their own workers according to their requirements. Apart from that, the SecureZone implementation has gone a much further by providing such workers. For example SecureZone is equipped with set of issuers that issue different kind of tokens. All these issuers are provided as abstract classes and application developers can extend these to customize them according to the application.

WS-Trust developers can contribute to the implementation by improving the set of issuers for different token types where different types of tokens are requested with different types of tokens used to identify the clients. This flexibility is not limited to issuers. Other workers (Renewers, Validators and Requesters) are also implemented similarly giving the same level of flexibility.

Handlers

WS-Trust implementation provides two handlers for Apache Axis. These two handlers are designed to overcome a limitation in Apache Axis implementation. When Axis is used as the SOAP engine the headers of messages are not visible at endpoints. Therefore two handlers were used at the client side and the server side. For example the STS Server Handler receives the request document and hands it over to STS Manager to build the response. This is the only task carried out by handlers.

To minimize the work at the client side in situations where WS – Trust is used by another Axis handler a special communicator was introduced along with the

STS Client Handler. In the case of WS – Secure Conversation usage scenarios these two components act to as an interface to WS-Secure Conversation when it is necessary to retrieve a security token. These components can be bypassed and a developer can handle the response and the request manually.

To ensure the integrity and the confidentiality of requests and responses WSDoAll handlers of the WS-Security implementation are being used.

The utility module

The utility module can be divided to two main parts.

1. Tokens

These tokens are being used by other components to build SOAP messages

2. Utilities functionalities

Utility functionalities are a set of common functionalities that are being used here and there of the implementation.

Trust Engine

The trust engine acts as a firewall for the targeted web service. It verifies the trust of an incoming request and allows trusted requests to be sent to the web service. An incoming request is subject to three levels of verification processes.

1. Verifies the tokens are sufficient to comply with the policies of the web service and that the message confirms to those policies.
2. Verifies the attributes of the claimant are proven by signatures.
3. Verifies that the issuers of the security tokens are trusted to issue the claims they have made.

This component is also independent of the SOAP Engine used and a developer can easily configure it to be used with any Java based SOAP engine.

5. Implementation

The WS-Secure Conversation and WS-Trust specifications are generic standards, and they are not specific to Axis or Java. Furthermore Axis was not developed with these specifications in mind. This made it challenge to map the specification requirements to the Apahce Axis. Therefore design and the implementation of the framework had to be completely done from the scratch to support Axis and to meet the standards of the specification.

Furthermore the team managed to develop core components of WS-Secure Conversation independent from Axis. Hence it would be an interesting experience to implement WS-Secure Conversation for other SOAP engines using the core components in the framework.

One of the major challenges faced while implementing WS-Trust is that the SOAP header is not visible to the web service. A workaround to face this problem had to be invented. The discovered solution is to implement a dummy trust service and the actual work is performed by the handlers.

6. Evaluation

SecureZone provides methods for establishing and maintaining secure conversation between communicating parties in accordance with the WS-Secure Conversation specification.

The framework is capable of maintaining the secure conversation by using deriving keys and it successfully exchanges the new key material and uses it according to the requirements of the developer.

The framework supports 2 out of the 3 security context establishment methods given in the specification. It supports for an SCT to be created by one of the communicating parties and propagated with a message. Here the sender is trusted to always create a new SCT. In situations where the service needs to establish trust, in other words when the service does not directly trust all requesters, this framework provides the necessary facilities to establish trust between the communicating parties as specified by WS-Trust. The third scenario specified in the specification which is “Establishing SCT using negotiation, exchanges” is not yet supported in the framework. The used WS-Security implementation is not yet mature enough to implement this functionality.

Benchmark test

A benchmark test was carried out to evaluate the performance gain provided by SecureZone against WS-Security implementation at Apache foundation’s “wss4j” project.

When using WS-Secure Conversation it requires additional communications for establishing the SCT, prior to normal application exchanges. Therefore it was expected that initial handshake would take additional time. After establishing the SCT, based on the secret key associated with the SCT “derived keys” are generated and they are used for securing the messages using symmetric key cryptography. Hence it was expected that subsequent message exchanges would increase the overall performance.

The test was carried out by securing one hundred SOAP messages using WS-Security The same one hundred SOAP messages were secured using SecureZone under same conditions. Following results were obtained.

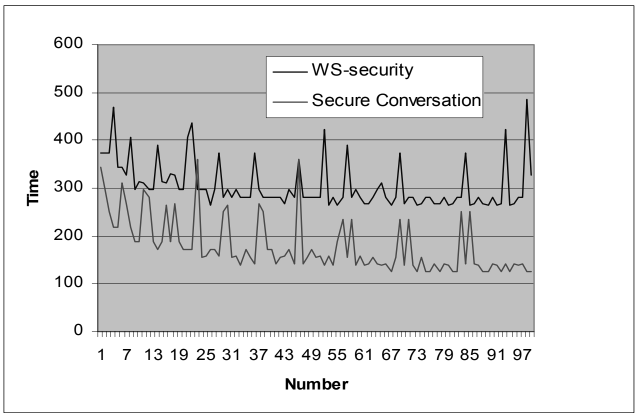

Figure 4: Test results (Upper line: WS-Security)

As expected WS-Secure Conversation has a performance improvement of almost double. As expected the initial exchange takes a longer time in SecureZone.

WS-Security also shows a similar behavior, i.e. the initial request takes a longer duration than the other number of messages. The reason for this is, WS-Security loads the X509 certificate in the initial request and keep it in memory. For the rest of the messages it uses the copy in the memory.

7. Comparison with existing technologies

The only publicly available implementation of WS - Secure Conversation and WS - Trust is Microsoft's Web Services Enhancements version 2.0 (WSE 2.0)[5]. This implementation caters only for the Microsoft .NET platform. Here the developer has the flexibility to manually work with the communication protocol where he/she is allowed to work with the security tokens and use them to encrypt/decrypt the required portions of the SOAP messages both at the client side and the server side. Also it is possible to provide automatic secure conversation feature to simply secure any web service by adding proper entries in the configuration files.

The java implementation provided in SecureZone is equally powerful as it provides the developer similar flexibilities in providing security. A developer can easily enable secure conversation on an Apache Axis web service with the interfacing components provided for Apache Axis (using ConversatonClientHandler and the ConversationServerHandler). Furthermore one can use the generic modules provided in SecureZone to extend the existing features or customize secure conversation protocols in securing web services. Also in the context of WS-Trust the SecureZone implementation is extremely extensible and flexible in terms of options available for the developers similar to what is available in WSE 2.0.

Furthermore the SecureZone implementation of WS - Secure Conversation and WS - Trust specifications is a part of Web Services Security for Java (WSS4J) project of Apache Web Services. Therefore with the contributions from a large number of experts, developers, quality assurance engineers available in the open source environment, it will continuously evolve to be a perfect implementation of the two specifications.

WSE 2.0 caters for Microsoft .NET web services only. With SecureZone implementation concentrated a lot on identifying components that are SOAP Engine independent. Therefore theoretically this implementation can be reused in any other Java based SOAP processing environment to provide WS - Secure Conversation and WS - Trust features. The WS - Security components of the WSS4J project were successfully tested with the BEA Weblogic application server since the core components of SecureZone were strictly based on the core components of WSS4J WS - Security implementation it is certain that the SOAP Engine independence is preserved in those components.

One major plus point about WSE 2.0 is that it provides a complete implementation of all the currently available specifications of the web services security stack. Since the availability of implementations such as WS-Policy it is possible to establish more complete usage scenarios. But in the context of Java SecureZone was unable to provide such functionalities, which are strongly dependant on the unimplemented specifications.

8. Conclusion

SecureZone successfully implemented the main features of WS – Secure Conversation and WS – Trust specifications. The architecture provided is highly flexible and the SOAP engine independence is preserved with the all the core components of the framework. These components were tested with Apache Axis SOAP Engine and they are yet to be tested for interoperability with Microsoft's WSE 2.0 for .NET framework which will be carried out once some of the other specifications in the web services security stack are implemented.

The SecureZone deliverables are now publicly available in the Apache WSS4J project and will be recognized as yet another valuable contribution to the world's leading open source software foundation, by a Sri Lankan team.

9. References

[1] WS - Secure Conversation Specification
ftp://www6.software.ibm.com/software/developer/library/ws-secureconversation.pdf
[2] WS - Trust Specification
ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf
[3] WS - Security Specification
ftp://www6.software.ibm.com/software/developer/library/ws-secure.pdf
[4] Web services Security For Java Project
http://ws.apache.org/ws-fx/wss4j/
[5] Web Services Enhancements (WSE)
http://msdn.microsoft.com/webservices/building/wse/default.aspx